Master Thesis

Language Identification using Deep Convolutional Recurrent Neural Networks

<Deutscher Titel bei englischer Arbeit>

Tom Herold

tom.herold@student.hpi.uni-potsdam.de

XX.02.2016

Supverisors
Prof. Dr. Christoph Meinel
Dr. Haojin Yang

Internet-Technologies and Systems
Hasso Plattner Institute
University of Potsdam, Germany

Abstract

Zusammenfassung

Acknowledgments

Blub

Contents

Abbreviations

LSTM long short-term memory

SGD stochastic gradient descent

LSTM

1. Introduction

1.1. Language Identification as the key to speech tasks

1.2. Contribution

1.3. Outline of the Thesis

2. The Language Identification Problem

2.1. Language Identification

2.2. Task Specification in This Thesis

3. Theoretical Background

3.1. Machine Learning

3.1.1. Types of Machine Learning

3.1.2. Classification

3.2. Building Blocks of Neural Networks

3.2.1. Fully Connected Layer

3.2.2. Convolutional Layers

3.2.3. Pooling Layers

3.2.4. Batch Normalization Layers?

3.2.5. Softmax Loss Function

3.3. Recurrent Neural Networks

3.3.1. Long Short Term Memory Networks

3.4. Hybrid Networks

- Convolutional Recurrent Neural Networks

3.5. Audio Representations

- MFCC
- Spectrogram
- Waveform

- Mel-scale

4. Related Work

4.0.1. A UNIFIED DEEP NEURAL NETWORK FOR SPEAKER AND LANGUAGE RECOGNITION

- [13]
- Almost identical: Deep Neural Network Approaches to Speaker and Language Recognition [12]
- high level overview of i-vector system
- Task: Language Recogniton + Speaker Recognition
- use bottleneck feature in the second to last layer
- input 7 static cepstra appended with 49 SDC
- DNN has 7 hidden layers of 1024 nodes each with the exception of the 6th bottleneck layer which has 64 nodes
- LRE11

4.0.2. EXTRACTING DEEP NEURAL NETWORK BOTTLENECK FEATURES USING LOW-RANK MATRIX FACTORIZATION

- [18]
- bottle neck feature improve classification results
- Task: Automatc Speech Recogniton (ASR)
- get bottleneck feature by low rank matrix factorization
- this is done by replacing the usual softmax layer weights by a linear layer with a small number of hidden units followed by a softmax layer
- BN laier is always last layer
- linear layer = FC without activation func

- uses DNN with 5 FCs with 1024 hidden units each + sigmoid activations + 1 BN layer
- softmax cross entropy loss
- 23 critical-band energies are obtained from a Mel filter-bank, with conversation-side-based mean subtraction = 150 dimensions
- further reduciton of output by PCA
- hybrid system of DNN + BN feeding into DNN + BN

4.1. LRE 2015

4.1.1. BAT System Description for NIST LRE 2015

- [11]
- participate in the "Fixed" and "Open" LRE Challenge
- segment data using automated Voice Activity Detection = previously trained NN
- 3042 segments (248 hours of speech) in train set and 42295 segments (146 hours of speech) in dev set.
- inputs 24 log Mel-scale filter bank outputs augmented with fundamental frequency features from 4 different f0 esti- mators
- used i-vector system

4.1.2. Discriminating Languages in a Probabilistic Latent Subspace

- [16]
- Probabilistic Linear Discriminant Analysis (PLDA) model
- In this paper, we review state-of-the-art generative methods, based on the Total Variability (TV) model [10], with the aim to improve their performance with discriminative fine-tuning of each language cluster at a time.

- TV maps audio into single low-dimensional vector, i-vector, that contains speaker, channel, and phonetic variability

4.1.3. Evaluation of an LSTM-RNN System in Different NIST Language Recognition Frameworks

- [17]
- used a one directional LSTM
- perform significantly better than i-vectors systems in LRE
- nice high level description of how LSTMs work
- inputs: random chunks of 2 seconds from which MFCC-SDC (Shifted Delta Coefficients)
- softmax cross entropy loss
- use last frame for scoring
- comparison if i-vector baseline to LSTM
- only used training data for 8 languages with more than 200hours of data
- US English (eng), Span- ish (spa), Dari (dar), French (fre), Pashto (pas), Russian (rus), Urdu (urd), Chinese Mandarin (chi)
- data split into 3, 10 and 30 seconds
- model: two hidden layers of 512 units followed by an output layer. The hidden layers are uni-directional LSTM layers while the output layer is a softmax with as many units as languages in the cluster
- LSTM os only better for short utterance (<= 10s)
- LSTM uses less parameters than i-vector

4.1.4. Frame-by-frame language identification in short utterances using deep neural networks

- [7]
- highlights the downsides/disadvantages of i-vector systems

5. Datasets

In this section we will explain the structure of our datasets, how we obtained them and what preprocessing steps are needed to extract our features.

Recent breakthroughs in deep learning were fueled by the availability of large-scale, well-annotated, public datasets, for example ImageNet [14]. Within the language identification community the TIMIT corpus of read speech [5] has long been the default test set. TIMIT contains a total of 5.4 hours, consisting of 10 sentences spoken by each of 630 speakers from 8 major dialect regions of the United States recorded at 16kHz. Given the short span of each individual sound clip, the overall corpus duration and limitation to one language it was necessary to obtain our data from somewhere else.

This thesis uses two primary datasets collected and processed by us. On the one hand we use speeches and statements from the European Parliament and on the other hand we rely on reports from news broadcasts sourced from YouTube.

5.1. Language Selection

5.2. EU Speech Repository

The EU Speech Repository is a collection of video ressources for interpretation students provided for free by the European Commission. The dataset is made up from debate of the European Parliament, commitee press conferences, interviews and tailor-made training material from EU interpretors. All audio clips are recorded in the speaker's native language and feature only one speaker.

With 131 hours of speech data it is the smaller of the two datasets. We obtained material in four languages: English, German, French, and Spanish

5.3. YouTube News Collection

Following the example of Montavon [10] we looked for large, public sources of speech audio. We first experimented with podcasts and radio stations, both of which are unsuited for the job. Podcasts usually feature only one speaker and radio contains a lot of noise in the form of music. From these initial insights we noticed that news broadcasts provided high quality speech audio data. To source a large variety of languages and gather enough hours of speech audio we sourced the majority of our data from YouTube.

YouTube Channel Name	Language
CNN	English
BBCNews	English
VOAvideo	English
DeutscheWelle	German
Euronewsde	German
N24de	German
France24	French
Antena3noticias	Spanish
RTVE	Spanish
VOAChina	Mandarin Chinese
Russia24TV	Russian
RTrussian	Russian

Table 1: YouTube Channel names and their corresponding language

For each target language we manually selected one or more YouTube channels of respected news outlets. For example for English we used the BBC and CNN to gather a variety of different accents. For a full list of channels refer to table 1. All channels were chosen regardless of their content, their political views or journalistic agenda.

Audio obtained from news coverage has many desired properties. The data is of high recording quality and hundreds of hours recording is available online. News anchors are trained to speak loud and clear, while still talking at a normal speed. News shows often feature guests or correspondents so we get a variety of speakers, which converse in a ordinary fashion with each other unlike speech audio obtained from reading texts aloud. Lastly, news shows feature all the noise one would expect from a real world situation: music jingles, non-speech audio from video clips and transitions between reports. The difficulty is also increased by mixed language reports. Many city, company, and personal names (e.g., New York City or Google for English) are pronounced in their native language and are embedded within the broadcast's host language. In essence we believe that speech data source from news broadcast represent an accurate, real-world sample for speech audio.

In contrast to the EU Speech Repository this dataset consists 1024 hours of audio data for the same four languages: English, German, French and Spanish. We also gathered

Feature	EU Speech Repository	YouTube News	YouTube News Extended
Languages	English, German, French, Spanish	English, German, French, Spanish	English, German, French, Spanish, Russian, Chinese
Total audio duration	4	4	6
Average clip duration	4	4	6
Audio Sampling Rate	4	4	6
Training Samples	4	4	6

Table 2: Comparison of the EU Speech and YouTube News dataset

an extended language set adding Mandarin Chinese and Russian to the dataset. The extended set is only used for the evaluation of the model extensibility as outlined in section 7.2.8. Table 2 provides a complete comparison between the datasets.

6. Implementation

This section will outline the employed software and hardware resources of the system, explain the data preprocessing, and describe the model architecture of our neural networks in detail.

6.1. Software

The language identification system is implemented in Python. We used the open source deep learning framework Keras[3] with the TensorFlow[1] backend for training our neural networks. Keras provides us with a set of higher level machine learning primitives such as convolutional layers and optimizing algorithms without abstracting too much away. Internally it builds on Google's open source numerical operation library TensorFlow which is optimized to quickly compute large multidimensional data on GPUs.

From a high level perspective, our system was designed in three parts.

The preprocessor Its job is to download, extract, and convert the data. The preprocessor clips the audio tracks for raw video footage and converts them into WAV files before ultimately converting the data into PNG images for the training. Details can be found in section 6.2.

The trainer

The evaluator

6.2. Data Preprocessing

All audio files have to be preprocessed before feeding them to the neural network. As a first step all files are encoded as uncompressed, lossless Waveform Audio File Format[1], WAVE, commonly know by its file extension *.wav. This conversion has two advantages: A lossless data codec allows for future audio manipulations with any quality loss and makes the data easily readable by third party programs and library such as SciPy[2].

Since our CNN does not operate on raw waveform audio signals we transfer it into the image domain. As introduced in section 3.5 we used a spectrogram representation of the audio file for model training. The spectrograms were generated using the open source

[1] `http://www.microsoft.com/whdc/device/audio/multichaud.mspx`, accessed 23.02.2017
[2] `https://www.scipy.org/`, accessed 23.02.2017

command line tool SoX[3]. Spectrograms are generated using a Hann window and 129 frequency bins along the frequency-axis (y-axis). The time-axis (x-axis) is rendered at 25 pixel per second. Each audio sequence is clipped into non-overlapping ten second segments. The final segment is discarded to avoid segments shorter than the required ten seconds. We decided against filtering silent section within the audio segment to keep the natural pauses between words and not disturb the regular speech rhythm. Frequency intensities are mapped to a gray scale. The resulting greyscale images are saved as lossless PNG files and are 500 pixel in width and 129 pixel in height. Appendix A includes the complete listing 1 for generating spectrogram images with SoX.

As can be seen in figure 1 the spectrograms feature very apparent bright ripple-like pattern. Each of these represents a strong activation of a certain frequency over time. Several frequency activations can be active at once constituting a particular phoneme or sound. A sequence of these phonemes forms words and is only interrupted by short speech pauses. It is our aim to learn the characteristic and unique composition of these frequency activation for every language in our classifier.

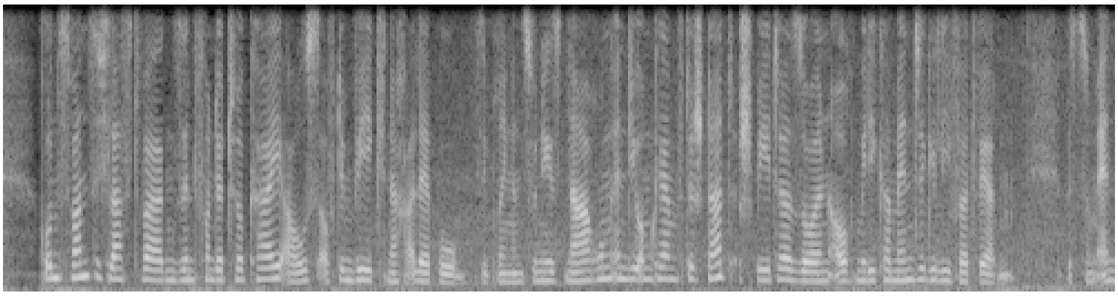

Figure 1: A spectrogram generated from a ten second German audio clip using SoX. Notice the bright ripple-like patterns. These frequency activations serve as the main features for the classifier.

- Spectrogram Generation
- Data Segmentation
- (Augmentation)
- train / test split
- number of training samples

[3]`http://sox.sourceforge.net/`, accessed 23.02.2017

6.3. CNN Architecture

- Layer Table
- Transfer Learning / Fine-tuning
- Variations

6.4. CRNN Architecture

- Layer Table
- Architecture Image
- Conv Features to time steps
- Transfer Learning / Fine-tuning
-

Layer Type	input maps	output maps	kernel	stride
Convolution with Batch Normalization	1	16	7x7	1
Max Pooling	16	16	2x2	2
Convolution with Batch Normalization	16	32	5x5	1
Max Pooling	32	32	2x2	2
Convolution with Batch Normalization	32	64	3x3	1
Max Pooling	64	64	2x2	2
Convolution with Batch Normalization	64	128	3x3	1
Max Pooling	128	128	2x2	2
Convolution with Batch Normalization	128	256	3x3	1
Max Pooling	256	256	2x2	2
Dropout				
Fully Connected	???	1024		
Fully Connected	1024	4		

Table 3: The layer architecture for the convolutional neural network CNN_A.

7. Experiments and Evaluation

In this chapter we show and discuss the results of training the outlined neural network architecture for spoken language identification. We introduce several performance metrics and present the results evaluated on our system. Further, we experiment with modified model architectures to optimize our model for noise robustness. To assess the real world performance of the neural network we augment our data to simulated various noisy environments. We show the classification performance of our approach by discussing the inter language discrimination and extensibility to other languages.

7.1. Setup

7.1.1. Hardware Resources

In order to facilitate Keras' and TensorFlow's hardware-accelerated computation we ran of training on CUDA[4] compatible GPU machines. All trainings and experiments were executed on two CUDA enabled machines belonging to the Internet Technologies and Systems chair. Details can be found in table 4.

	Machine A	Machine B
OS	Ubuntu Linux 14.04	Ubuntu Linux 16.04
CPU	Intel® Core™ i7-4790K @ 4GHz	AMD FX™-8370 @ 4GHz
RAM	16GB	32GB
GPU	Nvidia GeForce® GTX 980	Nvidia Titan X
VRAM	4GB	12GB

Table 4: Hardware resources used in training the neural network.

7.1.2. Data

For our performance evaluation we used the European Speech Repository and YouTube News dataset as describe in section 5. Both datasets were preprocessed and converted spectrogram images as described in section 6.2. Each spectrogram image represents a non-overlapping ten second duration of source audio file. We split both dataset into

[4] `https://developer.nvidia.com/cuda-zone`, accessed 30.01.2017

	European Speech Repository	YouTube News
Training Set	18.788	193.432
Validation Set	5.372	55.272
Test Set	2.684	27.632
Total	26.844	276.336

Table 5: The amount of samples for our training (70%), validation (20%) and testing (10%) set for the respective datasets.

a training (70%), validation (20%) and testing set (10%) and all files were distributed equally between all four language classes. The amount of samples per language is limited by language with least files to ensure an equal class distribution. The yield of the datasets could be increased by increasing the number of Spanish files. Nonetheless, the European Speech repository yields a total of 19.000 training images files which adds to roughly 53 hours of speech audio. The YouTube News dataset is considerable larger and yields a total of 194.000 training files, or 540 hours. Table 5 contains the detailed dataset splits.

Given the European Speech Repository's smaller size we only used it initially confirm the validity of our approach. Since we were satisfied with the results we did not do include it in the extensive robustness tests that we used for the evaluation on the YouTube News dataset. In addition to the original audio we augmented the news dataset with three different background noises to evaluate how well our model would hold out in non ideal, real world situations outside of a new broadcasting studio. For the first experiment we added generic white noise to data. For the second experiment we added noise to simulate an old phone line or bad internet connection during voice chat. Lastly, we added background music to the data. All experiments are described in detail below.

7.1.3. Training of the Neural Network Model

Neural networks have a multitude of hyperparameters that influence the training results drastically. In the following we will briefly explain our choice of hyperparameters and other important training settings.

Optimizer We employed an Adam[8] optimizer to quickly and efficiently convergence our model. The Adam solver utilizes momentum during gradient updates and to support a quicker convergence. We set the optimizer's parameters β_1 to 0.9, β_2 to 0.999, and ϵ to 1e-08. Overall we found it to be an order of magnitude quicker than using standard stochastic gradient descent (SGD). We resorted to SGD finetuning when we needed more control over the learning rate schedule and wanted smaller weight updates.

Weights Initializer All layer weights are initialized within the range is [0, 1) using Keras' default random Glorot uniform initializer[6].

Data Normalization The greyscale images are loaded using SciPy and normalized to the [0, 1] range. The shape for all inputs needs to be uniform across all samples and is set to [500, 129, 1], unless otherwise noted. The data loader uses Python generators[5] to keep the system's memory requirements low.

Learning Rate We set the initial learning rate to 0.001. Given the Adam optimizer's dynamic learning rate adaption we expect the learning rate to be increase or decreased after every epoch.

Batch Size We specified the batch size depending on the available VRAM of the training machine. We used a value of 64 for Machine A and 128 for Machine B. See section 7.1.1 for the hardware specifications.

Weight Regularization We employed the L2 norm as a weight regularizer for all convolutional and fully connected layers to improve the models generality. We penalize our loss with a weight decay value of 0.001.

Epochs We limited the model training to a maximum of 50 epochs when using the Adam solver. We usually reach convergence well below this threshold. For training sessions with SGD we increased this considerably. To speed up our workflow we employed an early stopping policy and stopped a training if the validation accuracy and loss did not increase within a ten window.

Metrics We observed the loss, accuracy, recall, precession, f1 measure, and equal error rate for both the training and validation set during model training. All values were saved to log files and visualized as graphs in Tensorboard[6].

Loss As is common for multivariate classification all models were trained with a softmax cross-entropy loss function.

[5] `https://docs.python.org/3/glossary.html#term-generator`, accessed 30.01.2017
[6] `https://www.tensorflow.org/how_tos/summaries_and_tensorboard/`, accessed 30.01.2017

7.2. Evaluation

7.2.1. Evaluation Metrics

In this section we discuss the evaluation metrics used throughout our experiments. All metrics are generally only defined for binary classification results. Given our multi-class prediction problem we will report the average of the individual class performance measure in the following sections.

Accuracy is a common measure in machine learning and is defined as the ration of correctly classified samples to all samples in the dataset. In the context of language identification this translates as:

$$Accuracy = \frac{\left|\{\text{correctly identified language samples}\}\right|}{\left|\{\text{all language samples}\}\right|}$$

Precision and Recall Precision defines the ratio of retrieved language samples that are correctly identified as belonging to said language. Recall is the fraction of correctly identified language samples to all samples belonging to this language.

$$truePositives = \left|\{\text{samples belonging to a language which were correctly identified as belonging to said language}\}\right|$$

fix set's curly braces

$$falsePositives = \left|\{\text{samples belonging to a language which were identified as belonging to another language}\}\right|$$

$$falseNegatives = \left|\{\text{samples belonging to a language which were incorrectly identified as not belonging to said language}\}\right|$$

$$precision = \frac{truePositives}{truePositives + falsePositives}$$

$$recall = \frac{truePositives}{truePositives + falseNegatives}$$

The F1 Score is the scaled harmonic mean of precision and recall. It is used to a have combined judgement of recall and precision, since one is generally not interested in assessing one without the other.

$$F1 = 2 * \frac{precision * recall}{precision + recall}$$

7.2.2. Results for EU Speech Repository Dataset

In order to verify our theoretic model of using convolutional neural networks for classifying audio data in the image domain we established a baseline with the smaller EU Speech Repository dataset. Previous work with CNNs showed that the careful arrangement of the neural network layers has a great effect on the final classification performance. If one does not use enough or sufficiently large layers the model is not able to properly distinguish between classes. Going to deep or using too large layer outputs increases the overall number of model parameters to a degree where both training time and classification performance suffers again. The goal of this experiment is find favorable settings for the number of convolutional layers needed, the kernel size of the convolutions, the number of output maps of the convolutional layer and finally the number of features of the fully connected layer.

For this particular dataset we tested three slightly different model architectures. Following the VGG-style model architecture of Simonyan et al. [2], we setup a first CNN with five convolutional layers as outlined in section 6.3. The first two convolutional layer use larger kernel sizes of 7x7 and 5x5 respectively. All the following kernels were set at 3x3. Each convolutional layer is followed by batch normalization and a 2x2 pooling with a stride of two. After the five convolutional blocks we add regularization through a fifty percent dropout layer before flattening all parameters to a fully connected layer with 1024 outputs. The final fully connected layer serves as a classifier outputting the prediction for our language identification. Henceforth, we will refer to this model as CNN_A.

A slightly adapted version called CNN_B has the same amount of convolutional layers but with a reduced number of feature maps. Instead of doubling the initial value of sixteen feature map for every convolutional layer we stick to 16 - 32 - 32 - 64 - 64 feature maps respectively. The fully connected layer has been reduce to only 256 output. Overall this model has significantly less parameters than the CNN_A. The purpose of this variation is to ensure that the proposed architecture for CNN_A is not unnecessarily complex.

Lastly we evaluated architecture CNN_C which uses constant kernel size of 3x3 for all convolutional layers. At the same time we increased the number of convolutional layers

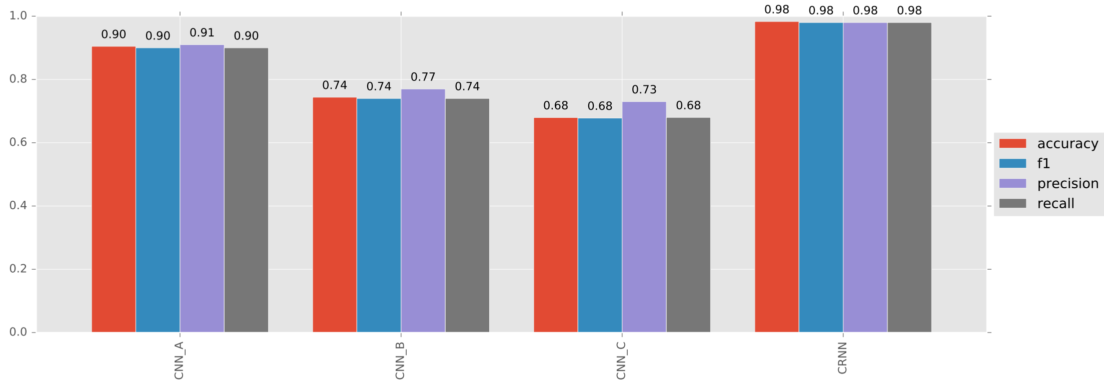

Figure 2: Performance measure comparison of three different CNN architectures evaluated on the EU Speech Repository dataset and our proposal of a CRNN model. CNN_A outperforms all other CNN approaches with a top accuracy of 0.90, but is bested by the CRNN's 0.98 accuracy, proving the potential of this thesis' approach.

to seven and extended feature maps for each layer: 64 - 128 - 256 -256 - 512 - 512 - 515. The remaining fully connected layers are identical to the CNN_A. The main difference here is the smaller receptive field of the convolutional layer which could negatively effect the model performance but should speed up the model training time overall.

CNN_A outperforms both of the other two network architectures with respect to all the evaluated performance measures, which can be seen in figure 2. With a top-1 accuracy of 0.904 it trumps CNN_B and CNN_C with 0.743 and ??? respectively. Comparing the F1 score we get a similar result: 0.90 versus 0.74 and ???. This experiment confirmed a few assumptions. Firstly, it proves that convolutional neural networks can be successfully used to classify audio data. Secondly, demonstrates that spectrogram images are meaningful representation for audio that retains enough information for language identification. Thirdly, it shows that large kernels for the initial convolutional layers are indeed favorable. The increased receptive field captures both the time domain and frequencies better. Based on these findings did some further testing with CNN_A. Based on Mishkin et al.[9] we switched the convolutional layers' ReLU activations to Exponential Linear Units[4] (ELU) but without any improvement. Baoguan et al. [15] propse to use 1x2 rectangular pooling windows instead of the conventional square ones. This tweak yields feature maps with larger width, hence longer features along the time domain. In theory this should increase the CNN's sensitivity at judging the occurrence of frequencies at certain time intervals. For this experiment we were unable to gain any improvement, but we will discuss this approach some more for our CRNN approach later.

The goal of this thesis is to evaluate the use of Deep Convolutional Recurrent Networks. Therefore we extended our previously best performing CNN_A with a bidirectional LSTM layer. We interpreted the CNN output as intermediate representation of the audio frequencies and used every vector entry along the x axis as a single step / input for the LSTMs. During training we froze the convolutional layer to only learn the frequency sequence of the audio sample. Our bidirectional LSTM layer trained two individual LSTMs with 512 outputs each, one training the input sequence from the start and one from the end. Both outputs were concatenated to form a single output vector with 1024 dimensions which is followed by single fully connected layer for classification. Our CRNN architecture outperformed all CNN approaches significantly. With a top-1 accuracy of 0.98 and a F1 score of 0.98 it proves the viability of the CRNN approach and reaffirms the central hypothesis of this thesis.

tabelle mit architecture CNN_C???

7.2.3. Results for YouTube News Dataset

7.2.4. Inter Language Discrimination

7.2.5. Effect of Audio Duration

7.2.6. Noise Robustness

7.2.7. Background Music Robustness

7.2.8. Model Extensibility

7.2.9. Visualization

7.2.10. Discussion and Comparison

- evaluierung nur auf 10s snippets nd nicht auf ganzen audio files - noch bessere perf mit majority voting über mehrere segmente

8. Conclusion and Future Work

8.1. Conclusion

8.2. Future Work

Appendices

A. Audio manipulation with SoX

```

sox -V0 input.wav -n remix 1 rate 10k spectrogram -y 129 -X 50 -m -r -o spectrogram.png



V0 - verbosity level
n - apply filter/effect
remix - select audio channels
rate - limit sampling rate to 10k; caps max frequency at 5kHz according to Nyquist-Shannon sampling theorem
y - spectogram height
X - pixels per second for width
m - monochrome output
r - disable legend
o - output file
```

Listing 1: Generating monochrome spectrograms with SoX

```
scale_factor1=0.94;
scale_factor2=0.05;

sox -m \
    -v $(echo "$(sox input.wav -n stat -v 2>&1) * ${scale_factor1}" | bc -l) input.wav \
    -v ${scale_factor2} <(sox input.wav -p synth whitenoise) \
    -b 16 output.wav
```

Listing 2: Adding white noise to an audio file

References

[1] Abadi, Martín ; Agarwal, Ashish ; Barham, Paul ; Brevdo, Eugene ; Chen, Zhifeng ; Citro, Craig ; Corrado, Greg S. ; Davis, Andy ; Dean, Jeffrey ; Devin, Matthieu u. a.: Tensorflow: Large-scale machine learning on heterogeneous distributed systems. In: *arXiv preprint arXiv:1603.04467* (2016)

[2] Chatfield, K. ; Simonyan, K. ; Vedaldi, A. ; Zisserman, A."ť: Return of the Devil in the Details: Delving Deep into Convolutional Nets. In: *British Machine Vision Conference*, 2014

[3] Chollet, François: *Keras*. `https://github.com/fchollet/keras`, 2015

[4] Clevert, Djork-Arné ; Unterthiner, Thomas ; Hochreiter, Sepp: Fast and accurate deep network learning by exponential linear units (elus). In: *arXiv preprint arXiv:1511.07289* (2015)

[5] Garofolo, John S. ; Lamel, Lori F. ; Fisher, William M. ; Fiscus, Jonathon G. ; Pallett, David S.: DARPA TIMIT acoustic-phonetic continous speech corpus CD-ROM. NIST speech disc 1-1.1. In: *NASA STI/Recon technical report n* 93 (1993)

[6] Glorot, Xavier ; Bengio, Yoshua: Understanding the difficulty of training deep feedforward neural networks. In: *Aistats* Bd. 9, 2010, S. 249–256

[7] Gonzalez-Dominguez, Javier ; Lopez-Moreno, Ignacio ; Moreno, Pedro J. ; Gonzalez-Rodriguez, Joaquin: Frame-by-frame language identification in short utterances using deep neural networks. In: *Neural Networks* 64 (2015), S. 49–58

[8] Kingma, Diederik ; Ba, Jimmy: Adam: A method for stochastic optimization. In: *arXiv preprint arXiv:1412.6980* (2014)

[9] Mishkin, Dmytro ; Sergievskiy, Nikolay ; Matas, Jiri: Systematic evaluation of CNN advances on the ImageNet. In: *arXiv preprint arXiv:1606.02228* (2016)

[10] Montavon, Gregoire: Deep learning for spoken language identification. In: *NIPS Workshop on deep learning for speech recognition and related applications*, 2009, S. 1–4

[11] Plchot, Oldrich ; Matejka, Pavel ; Fér, Radek ; Glembek, Ondrej ; Novotný, Ondrej ; Pešán, Jan ; Veselý, Karel ; Ondel, Lucas ; Karafiát, Martin ; Grézl, František u. a.: Bat system description for nist lre 2015. In: *Proc. Odyssey*, 2016

[12] Richardson, Fred ; Reynolds, Douglas ; Dehak, Najim: Deep neural network approaches to speaker and language recognition. In: *IEEE Signal Processing Letters* 22 (2015), Nr. 10, S. 1671–1675

[13] Richardson, Fred ; Reynolds, Douglas ; Dehak, Najim: A unified deep neural network for speaker and language recognition. In: *arXiv preprint arXiv:1504.00923* (2015)

[14] Russakovsky, Olga ; Deng, Jia ; Su, Hao ; Krause, Jonathan ; Satheesh, Sanjeev ; Ma, Sean ; Huang, Zhiheng ; Karpathy, Andrej ; Khosla, Aditya ; Bernstein, Michael ; Berg, Alexander C. ; Fei-Fei, Li: ImageNet Large Scale Visual Recognition Challenge. In: *International Journal of Computer Vision (IJCV)* 115 (2015), Nr. 3, S. 211–252. http://dx.doi.org/10.1007/s11263-015-0816-y. – DOI 10.1007/s11263–015–0816–y

[15] Shi, Baoguang ; Bai, Xiang ; Yao, Cong: An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2016)

[16] Sizov, Aleksandr ; Lee, Kong A. ; Kinnunen, Tomi: Discriminating Languages in a Probabilistic Latent Subspace.

[17] Zazo, Ruben ; Lozano-Diez, Alicia ; Gonzalez-Rodriguez, Joaquin: Evaluation of an LSTM-RNN System in Different NIST Language Recognition Frameworks. In: *Odyssey 2016* (2016), S. 231–236

[18] Zhang, Yu ; Chuangsuwanich, Ekapol ; Glass, James R.: Extracting deep neural network bottleneck features using low-rank matrix factorization. In: *ICASSP*, 2014, S. 185–189